Which of the following reasons make programming OpenFlow switches difficult?



Networks often have switches from multiple hardware vendors.



**Network policies involve many independent tasks that may have conflicting behaviors**.



OpenFlow networks require a single, centralized network controller.



**An OpenFlow controller only sees events for packets that the switches do not know how to handle.**

****

**OpenFlow switches control switches at a low level of abstraction.**

2.Question 2

What are some examples of tasks that a northbound API can perform?



Offering a standard API to program switches from multiple different vendors, thus providing forwarding independence.



**Ensuring that OpenFlow rules are installed onto a switch in the correct order.**

****

**Performing optimizations to flow-table rule installations to ensure correct network behavior.**

****

**Computing the k-shortest paths between different nodes in a network.**



Chaining actions together to allow multiple actions to be performed for a single flow-table entry.

1 point

3.Question 3

Suppose that a network operator wants to express a policy that drops all traffic except for traffic from a specific IP prefix (i.e., group of IP addresses). What problems exist with expressing this policy as a set of OpenFlow rules?



**Switches have limited room for flow table entries and cannot store rules for all IP addresses in a prefix by default.**

****

**If the controller installs the rules in the wrong order, all traffic might be dropped, instead of allowing the permitted traffic.**

****

**If the operator starts adding more IP addresses that are permitted to send traffic, performance of the switch might suffer as the number of entries in the switch table grows.**

****

Flow table entries can only perform exact matches on IP addresses, so installing rules that operate on groups of IP addresses will take many flow-table entries in the switch**.**

****

**The controller may see more than the first packet for a flow that it has already installed.**

1 point

4.Question 4

Which of the following have examples of sequential composition?



Block all port 80 traffic and count the number of packets that are blocked.



**Block all traffic except port 80 traffic, and load balance port 80 traffic across three destination IP addresses.**

****

**Block all traffic except port 80 and count the number of packets that are not blocked.**

****

Block all traffic except port 80 and count the number of packets that are blocked.



**Load balance traffic based on whether the traffic matches a particular destination IP prefix.**

1 point

5.Question 5

Which of the following have examples of parallel composition?



Perform MAC learning on all traffic that is not blocked by a firewall rule.



**Perform load balancing on any traffic that arrives from prefix 130.207.0.0/16 or is destined to port 80.**

****

**Count all port 80 traffic.**



Load balance traffic across multiple destination IP addresses and perform routing to the resulting destination IP addresses.



**Block all traffic whose source IP address is contained in a list of blocked IP addresses.**

1 point

6.Question 6

Which of the following is the appropriate Pyretic rule for matching traffic that sends packets from either 10.0.0.1 or 10.0.0.2 to IP address 1.2.3.4?



(match(srcip=10.0.0.1) + match(srcip=10.0.0.2)) + mod(dstip=1.2.3.4)



(match(srcip=10.0.0.1) >> match(srcip=10.0.0.2)) + mod(dstip=1.2.3.4)



(match(srcip=10.0.0.1) >> match(srcip=10.0.0.2)) >> mod(dstip=1.2.3.4)



**(match(srcip=10.0.0.1) + match(srcip=10.0.0.2)) >> mod(dstip=1.2.3.4)**



match(srcip=10.0.0.1) >> (match(srcip=10.0.0.2) >> mod(dstip=1.2.3.4))

1 point

7.Question 7

Which of the following is the appropiate Pyretic rule for sending traffic from source IP address 10.0.0.1 to destination IP address 10.1.2.3 and traffic from source IP address 10.0.0.2 to destination IP address 10.2.3.4?



(match(srcip=10.0.0.1) + mod(dstip=10.1.2.3)) >> (match(srcip=10.0.0.2) + mod(dstip=10.2.3.4))



**(match(srcip=10.0.0.1) >> mod(dstip=10.1.2.3)) + (match(srcip=10.0.0.2) >> mod(dstip=10.2.3.4))**



(match(srcip=10.0.0.1) >> mod(dstip=10.1.2.3)) & (match(srcip=10.0.0.2) >> mod(dstip=10.2.3.4)) not answer



(match(srcip=10.0.0.1) + mod(dstip=10.1.2.3)) + (match(srcip=10.0.0.2) + mod(dstip=10.2.3.4))



(match(srcip=10.0.0.1) >> mod(dstip=10.1.2.3)) >> (match(srcip=10.0.0.2) >>mod(dstip=10.2.3.4))

1 point

8.Question 8

Which of the following might be examples that would use a dynamic policy?



**Check each new traffic flow against a list of permitted and prohibited source IP addresses that may change over time**.



**Assign each new traffic flow to a different destination IP address based on a round-robin load balancing policy across servers.**



Drop all traffic arriving from source IP addresses in prefix 10.0.0.0/8.



**Forward traffic through the network based on shortest paths routing, re-routing traffic appropriately when links or nodes fail.**



Count all traffic arriving from source IP addresses in prefix 10.0.0.0/8.

1 point

9.Question 9

How often does the update\_policy() function get called?



For the first packet of every new flow arriving from any host.Not Answer



For every arriving packet from any host.



For every first packet arriving from any host. MAY Be The Answer



For every packet arriving from hosts whose MAC addresses are in the firewall.Not Answer



For every packet arriving from hosts whose MAC addresses are not in the firewall.

1 point

10.Question 10

The code in the link from the pre-amble is intended to implement a firewall that allows traffic from source MAC addresses 1 and 2 and drops all other traffic. Upon running the code, however, you will notice that the behavior is not as expected. (Or, you might read it to figure that it will not work!) What behavior will result from running the code referred to in the link?



**Some traffic from hosts on the network other than hosts with MAC address 1 and 2 may be allowed**.



All traffic will be allowed.



**Some traffic from hosts 1 and 2 may be dropped.**



All traffic will be dropped.



Traffic from MAC address 1 will be allowed, but traffic from MAC address 2 will be dropped.